UDK: 626/628:33/628.1
Originalni naučni rad

ANALYSING THE UNCERTAINTY OF FLOODING CAUSED BY PIPE BURST AT CITY SCALE

M. GUIDOLIN, A. S. CHEN, E. C. KEEDWELL, Slobodan DJORDJEVIĆ, Dragan A. SAVIĆ
Centre for Water Systems, College of Engineering, Mathematics and Physical Sciences, University of Exeter, Harrison Building, North Park Road, Exeter, EX4 4QF, UK

ABSTRACT

To evaluate the risk of potential pipe failure in a city, we developed a fast flood modelling framework that can automatically analyse the terrain data to set up the model inputs for each simulation. Thus, multiple scenarios covering a wide range of possible pipe burst locations can be simulated to identify the likelihood of an area being affected. The results will help the water company understand the risk and identify hotspots that are vulnerable to pipe failure such that a better asset management plan can be achieved to minimise the risk from their own buried assets.

Keywords: Pipe failure, Citywide, Framework, 2D inundation model, CADDIES, GPU

1. INTRODUCTION

Performing a risk assessment of potential flooding due to pipe failure events on a citywide scale is a challenging and time-consuming task due to the large uncertainty of the locations and the discharge of bursting pipes or collapsed and blocked sewers. A comprehensive analysis would require a huge number of hydraulic simulations to cover the whole possibility of scenarios. The time pressure is further increased if an accurate two-dimensional (2D) flood model is required.

This paper presents an advance software framework designed to quickly perform a large number of 2D flood simulations on a modern desktop computer. This framework is composed of three main components: 1) the reduction of the computation load for each pipe failure location, 2) the use of a new rapid 2D inundation model and 3) the speedup of executions using advance parallel techniques.

In order to reduce the computational load, a novel technique has been implemented to delineate the sub-catchments for modelling from a large catchment of the city. The terrain data for each possible pipe-failure location is automatically extracted from the large citywide terrain model catchment. The number of cells of the sub-catchment terrain is further reduced by an automated procedure that uses the ground characteristics of the area. The framework uses one of the CADDIES 2D [1] inundation models to rapidly simulate flooding resulting from pipe-burst events, which adopts simple transition rules for modelling, based on a cellular automata technique, instead of solving complex Shallow Water Equations (SWE). The new approach improves modelling efficiency without compromising accuracy. The model further reduces the computational load using a dynamic domain that automatically grows from the single cell that the failure occurs to the extent that the water reaches.

The simplified feature of cellular automata allows the CADDIES 2D model to be easily implemented in parallel computing environments. Thus the model can use not only the multiple cores of modern CPU but also the massive parallelism of the GPU on modern graphics cards. The framework presented in this work takes advantage of this possibility to automatically execute multiple simulations in parallel on any graphics card or multi-code core CPU installed on the desktop machine.

This new framework has been applied to two UK cities with commercial sensitive real-life pipe failure data using high resolution terrain data. The results obtained demonstrate that the new framework can be quickly assess potential flood risk due to pipe burst or sewer failure events on a modern desktop computer.

2. FRAMEWORK

Figure 1 shows the structure of execution flow of the framework. One of the main objectives of the framework is to reduce the computational load for each pipe failure simulation. Thus its first step is to execute the "sub-catchments creator" process which selects the sub-catchments, centred to the pipe failure locations, from the citywide catchment data as the modelling domain. The size of the reduced domain is an input parameter to be determined by users. The other inputs for the process include the full digital elevation model (DEM) of the catchment, and the list of pipe failure events with the location and the inflow information. The process produces the reduced squared DEM and simulation configuration file of each event for hydraulic modelling.

This process has an extra feature which is used to further reduce the computational load for the 2D inundation model. Assuming that the water from the pipe failure location is unlikely to travel to upstream, the cells with elevations that are higher than the elevation of the cell with failure plus a given threshold value are futher excluded from modelling.. This threshold is used to reduce the risk of not simulating the flooding of upstream cells when an event that is located at the bottom of a pond that has a very large inflow. The value of this threshold can be passed as input parameter.

After the "sub-catchments creator" process is executed, the next step of the framework is to execute the "simulation queue" process which runs the 2D overland inundation model for each event. The model used in this work is Weighted Cellular Automata 2D (WCA2D), which is a part of CADDIES 2D models. This model improves the methodology used in the CA2D model [1].

The WCA2D is a diffusive-like model that ignores inertia terms and momentum conservation and it uses a cellular automata (CA) approach. The CA technique offers a versatile method for modelling complex physical systems using simple operations [2]. This simplification dramatically reduces the computational load compared to a physically based model. The simple operations that govern the evolution of each cell's state make use of the previous state of the cell itself and of those in its neighbourhood. Since computing the new state of a cell does not depend on the state of any other cells at the same time step, CA algorithms are well suited to parallel computation.

The WCA2D model has been implemented using the CADDIES Application Programming Interface (API) framework [3]. The CADDIES API defines a standard set of methods, data structures and variables that can be used to develop parallel CA algorithms. The main idea of the CADDIES API is that a developer needs to write the code of the CA model only once. After that, the CADDIES API gives the flexibility to produce the same CA model for different high performance acceleration techniques without changing the code or with minimum effort. Thanks to the CADDIES API, the WCA2D model can be executed in a multi-core CPU using OpenMP library [4] and in a multi-core CPU and on a graphics card GPU using the OpenCL library [5].

The implementation of the WCA2D model used in this paper, called CADDIES-caflood, is publicly available using an open source license in the website of the Centre for Water Systems at: http://cws.exeter.ac.uk.

Since the WCA2D model can be executed on either CPU or GPU and the events are independent from each other, the "simulation queue" process can execute multiple events in parallel if multiple computational resources are available on the machine. The parallel technique used is a master/worker pattern and it is implemented using gnu parallel [6]. The "simulation queue" process represents the master which coordinates the tasks/events to be executed, while the various CPUs/GPUs available in the machine represent the workers where the tasks are executed. The master contains the list of tasks that are ready or are completed and submit the next task to be executed to next free worker that finished the previous task. This technique has the advantage that if a desktop machine has multiple CPUs and/or multiple GPUs these can be easily added to the pool of workers. Furthermore, with a master/worker patter is usually straight forward to also add the computational resources available on any remote machine if needed. A drawback of using multiple resources simultaneously is that the various computations could add an overhead to the shared resources, i.e. the memory-bus and the CPU cores. This overhead influences the run-time of a single simulation; however, this should have a minimal impact to the total run time in comparison of the time gain obtained due to the parallel executions. The final step of the "simulation queue" process is to automatically archive the results of each simulation into a single zip file.

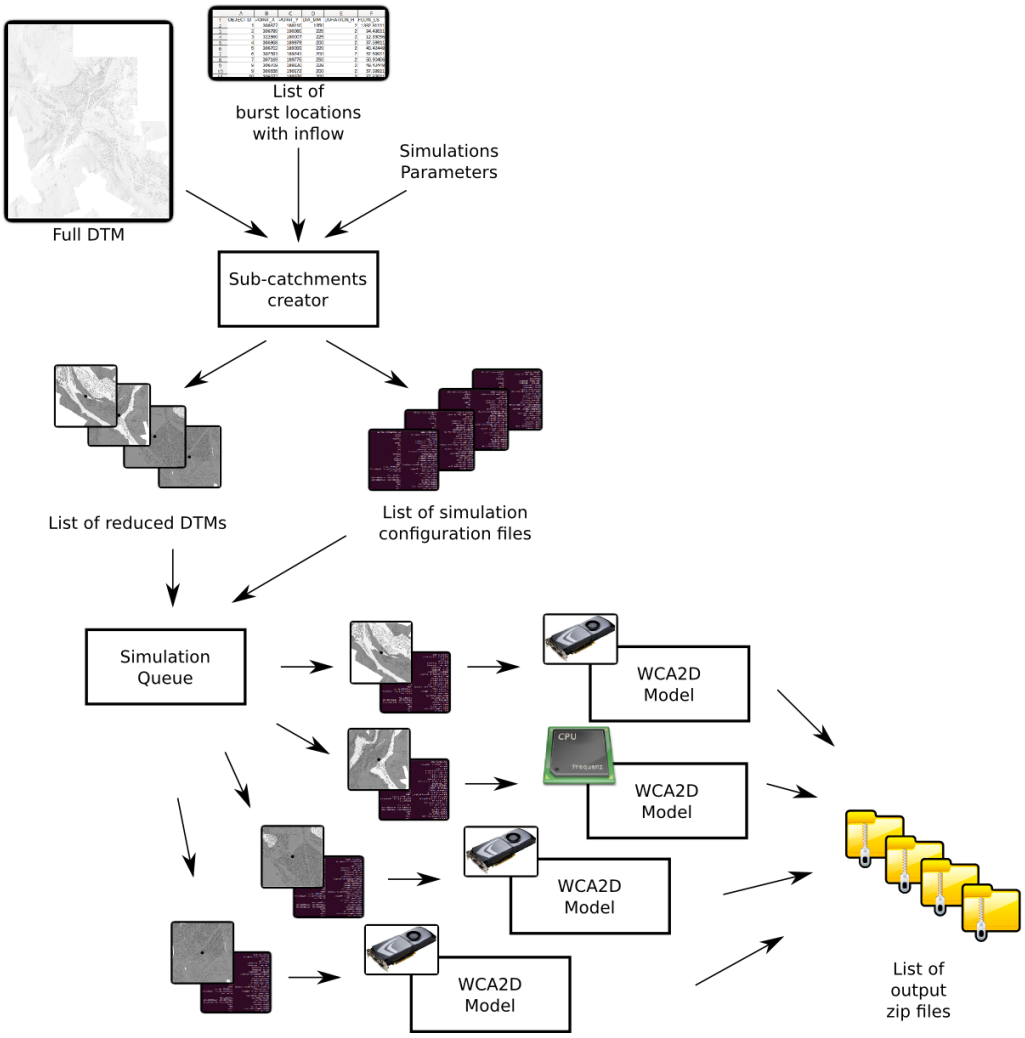

Figure 1: Structure of execution flow of the framework

3. TEST CASE

The framework developed in this work has been applied to a test case composed of around 500 pipe failure locations in two large UK cities. The first city is a highly urbanised region with large areas of mild slope while the second one is a large village in a valley with steep slopes. The terrain elevation of the first city ranges from 0 to 115 metres and the average slope without buildings is 6.5%. The second city has a terrain elevation ranging from 37 to 313 metres and the average slope without buildings is 26.3%. The digital elevation models (DEM) of the two cities, obtained from LiDAR, cover respectively an area of 63 and 30 square kilometres. The DEMs have a 2m resolution and thus they are composed respectively of around 13 million and 5 million data cells.

The two catchments contain rivers, channels and the sea. However, for simplicity they were not modelled separately and thus the elevation retrieved by the LiDAR was used directly. In the DEMs, the area covered by the building was elevated by a fixed amount, 10 metres, and a constant Manning roughness of 0.02 was used to the whole domain during the simulations. Figure 1 show the two full DEMs with the locations of the pipe failure events. In this figure the accuracy of the location points has been intentionally degraded owing to commercial sensitivity.

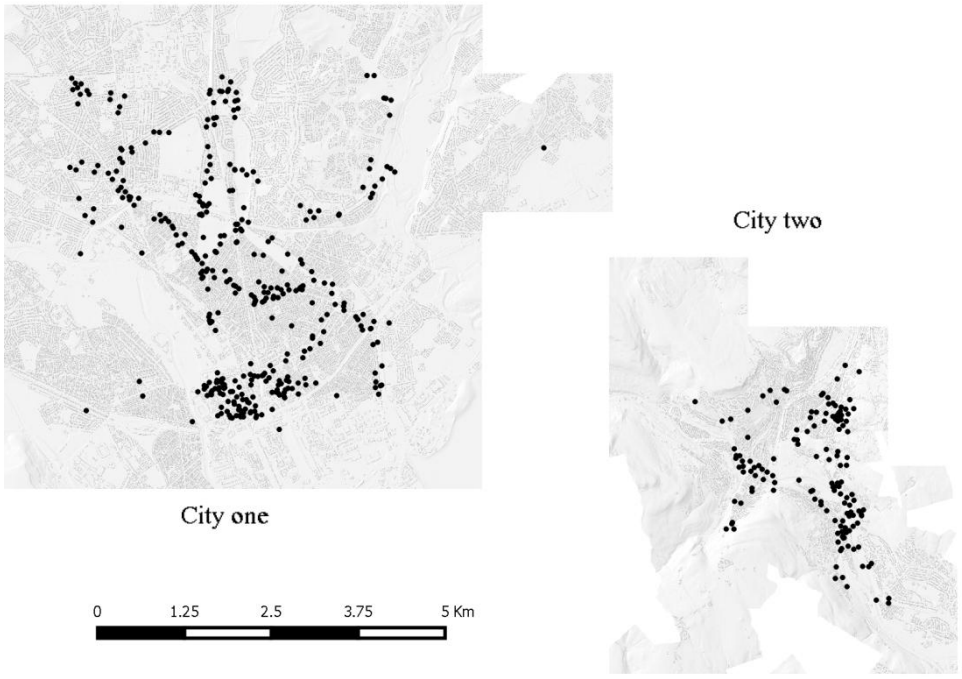

Figure 2: The DEMs of the two UK cities with the pipe burst location points.

For each location a single specific inflow was applied and all the inflows were set to have a fixed duration of two hours. The maximum inflow between all the events was around 2 m^3/s and the average inflow was around 0.187 m^3/s. The modelling of the drainage system was not included since this is not implemented in the WCA2D model. For each pipe failure event, the full simulation time was 2 hours and the outputs produced were the maximum inundation depth and maximum velocity.

Four events, two from the first city and two from the second one, were used in order to analyse the run times obtained by using different computational resources and the various features of the framework. The inflow values for these four events are: a) 1.382 m^3/s, b) 0.546 m^3/s, c) 0.547 m^3/s, and d) 0.355 m^3/s.

The maximum inundation depth results produced by the WCA2D model for the four events were compared visually with the results obtained by a simple rolling ball algorithm. This algorithm identifies the downstream flow paths by rolling a virtual ball using the terrain information. When the ball reaches a possible pond area, the depression gets filled until the water level reaches the elevation of an outlet point where the ball can resume rolling. The volume added to the pond is removed from the total volume produce by the inflow. The algorithm stops when all the inflow volume has been used. The rolling ball algorithm is useful for identifying quickly possible flood areas since it usually take few seconds to execute. Thus it is commonly used for performing pipe failure analysis of many events. However, it does not produce accurate results since only one single outlet is identified for each a pond, neither the flow dynamic which could change the direction of flood propagation. The results produced by the WCA2D were also compared visually with the results produced by InfoWorks ICM 3.0 [7]. This software is widely used in the water industry in the UK [8] and it uses a fully 2D hydrodynamics model. InfoWorks was considered a good benchmark to test the accuracy of the new model. However, unlike WCA2D, InfoWorks uses an irregular triangular mesh grid for the computation and computes the inertial term of the SWEs. Thus some discrepancy between the results of InfoWorks and WCA2D model is expected.

The framework was executed using an elevation threshold of 5 metres, i.e. all the cells that had an elevation without building higher than the pipe failure cell elevation plus 5 metres were excluded from modelling. For each failure location, the squared sub-catchment created had a side length of 2000 metres

centred to the location, i.e. the reduced squared DEM covered an area of 4 square kilometres for a maximum of 1 million cells. Figure 2 shows in grey the data cells of the four reduced DEMs generated by the framework for the four events. It is possible to see that mainly in the second city, which has a terrain with high slopes, a large part of the two DEMs are composed of no data cells upstream of the inflow cell.

The execution of the framework and all the simulations were performed on a desktop machine with an Intel Core i7-4770K CPU having four physical and eight virtual cores at 3.50GHz, 32GB of main memory and a Tesla K20c graphics card with 2496 CUDA cores and 5GB of video memory.

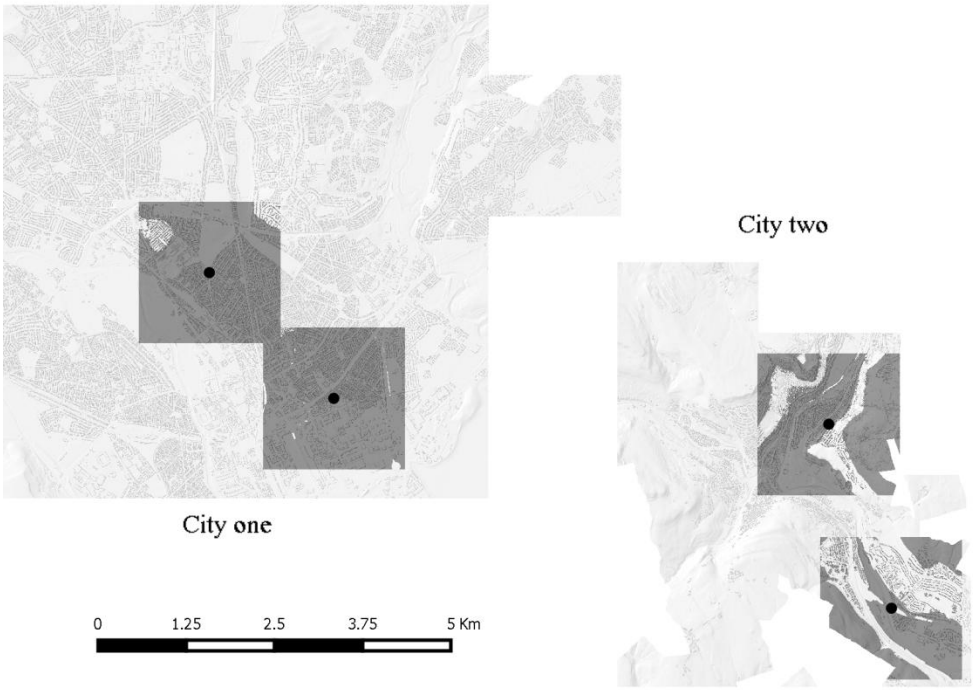

Figure 3: The reduced DEMs (in dark grey the data cells) of the sub-catchments generated by the framework for the four events analysed.

4. RESULTS

Figure 4 shows the cells identified as flooded by the rolling ball algorithm, and the contour at 0.1m of the maximum inundation depth results produced by WCA2D model and InfoWorks ICM. It is possible to see that the extent of the inundation at 0.1m is very similar between the two models. There are some small differences, mainly in the results of pipe failure events (a) and (b) between narrow spaces between rows of buildings. These differences could be caused by the use of an irregular triangular mesh in InfoWorks ICM. The creation of the mesh could change the elevation of the access points of the narrow spaces. The comparison between InfoWorks ICM and WCA2D in these four events shows that while the WCA2D model is a simplified model, it can produces results that are equivalent to a fully hydrodynamics model that compute the full SWEs.

The extent of the inundation identified by the rolling ball algorithm largely differs from the one identified by the other two models. This simple algorithm over predicts the flood extent in a single direction. This is visible for the event (a) and (b) where the terrains have mild slopes. The over prediction in one direction is less visible for the events in the second city where there are large slopes; but for example in event (d), the algorithm does not identify the area with many buildings near the failure location as flooded, which were caused by the secondary direction of flood movement.

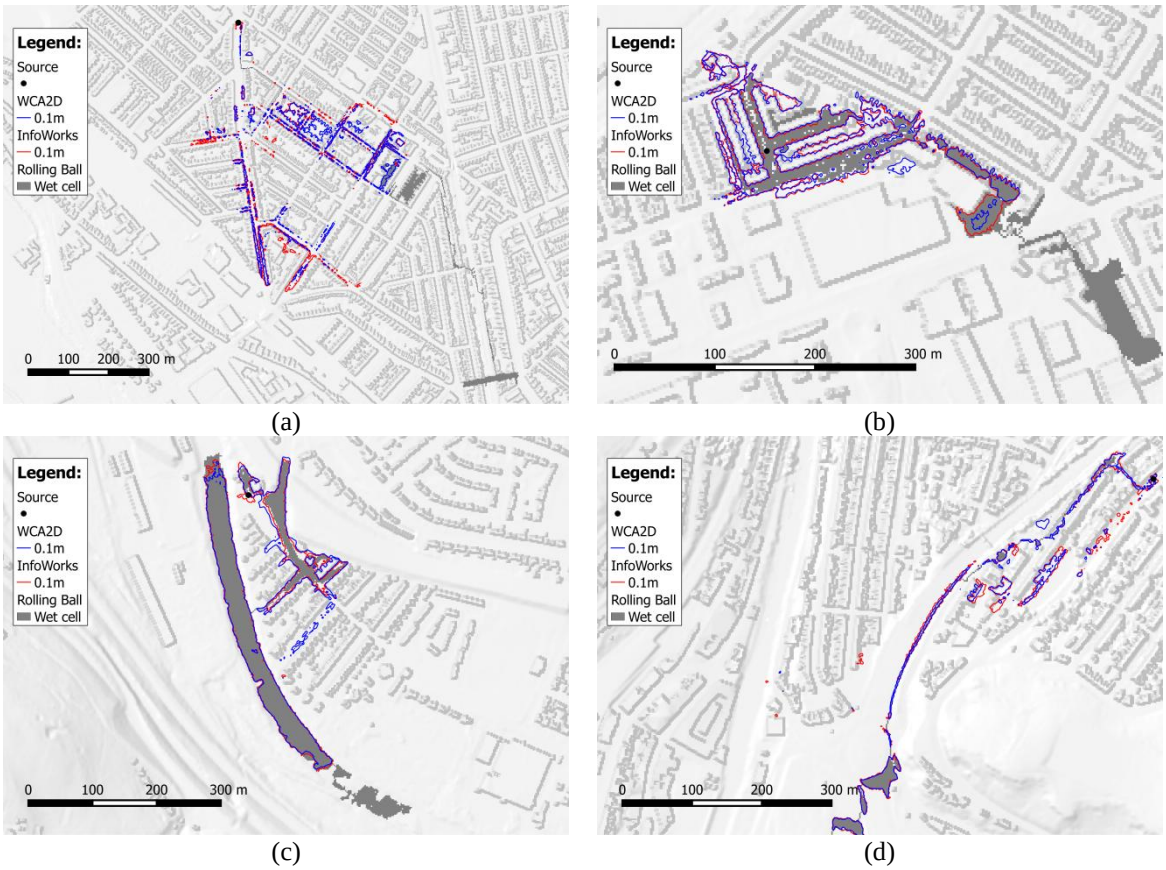

Figure 4: Cells identified as wet by the rolling ball algorithm (grey cells) and the contours at 0.1m of the maximum inundation depth results produced by WCA2D model (blue lines) and InfoWorks ICM (red lines)

Table 1 shows for each events the run times obtained by executing the full domain and the reduced domain simulations, i.e. respectively using the original citywide DEM and using the reduced DEM with the expanding domain algorithm. Furthermore, it shows the run times obtained running the simulations on the CPU using the 8 virtual cores and on the GPU. The table shows also the speed ups obtained between the different configurations.

It is possible to see that when the simulations are executed on the 8 cores of the CPU and using the DEM of the full domain, the run times are from almost one to almost two hours. Table 1 shows the performance advantage of reducing the computational load for each simulation. The execution on the CPU of the simulation with the reduced DEM and the expanding domain produces run times that are from 9 to 35 times faster.

Table 1: Run times and speed ups on various simulation configurations and hardware for the four events

	Event											
	(a)			(b)			(c)			(d)		
Simulation	Run time (s)		Speed up	Run time(s)		Speed up	Run time (s)		Speed up	Run time (s)		Speed up
	CPU	GPU		CPU	GPU		CPU	GPU		CPU	GPU	
Full domain	5193	576	9.0	2860	355	8.1	3255	368	8.8	6833	720	9.5
Reduced domain	223	27	8.3	83	10	8.3	237	21	11.3	707	79	9.7
Speed up	23.3	21.3		34.5	35.5		13.7	17.5		9.7	9.1	

The impact of the high computational power of the GPU is also visible in Table 1 where the same simulations are from 8 to over 9 times faster when executed on the graphics card. Finally, the run times of the four simulations using the reduced computational load and the parallel power of the GPU are from 10 second to just over a minute. These are up to 280 time fasters than the full simulation on the CPU.

Another advantage of creating a sub-catchment for each pipe burst event is that also the memory used during the simulation is reduced. For example, around 1GB of memory is used for simulating an event using the full DTM of the first city. While only around 130MB is used for simulating any event using a reduce DTM. Given that the graphics card used for this test case has a large amount of memory and there are many parallel cores available; running multiple events simultaneously on the GPU instead of only one single event at time could further reduce the total run time.

Table 2 shows that the "sub-catchments creator" process took just over four minutes to execute. This is in average only half a second for each simulation. Table 2 shows also a summary of the total run time obtained for the test case presented using three configurations on the "simulation queue" process. These configurations are: 1) all the events are simulated one at time on the 8 virtual cores of the CPU; 2) all the events are simulated one at time on the GPU; and 3) the events are executed in parallel in either the CPU or GPU. In this last configuration, in order to avoid overloading the main processor, the CPU simulations are performed using only 4 cores. Furthermore, two events can be executed simultaneously on the GPU.

Running all simulations sequentially on the CPU took over half a day, while it only required less than two hours to do so on the GPU. Finally, by running the events in parallel on either the CPU or GPU, it took around 88 minutes to execute around 500 pipe failure events for two large cities.

Considering that it took in average just over 10 seconds to simulate an event, it would be possible to simulate more than eight thousand pipe failure events in a day using this framework on a modern desktop machine.

Table 2: Summary of the test case run timesExecution type

	Run time		
	CPU	GPU	CPU+GPU
Sub-catchments creator		4.33 min	
Framework total time	883.34 min	107.60 min	88.15 min
Maximum of all events	706.9 sec	80.2 sec	603.8 sec
Minimum of all events	11.1 sec	3.2 sec	4.7 sec
Average between all events	106.0 sec	12.4 sec	10.6 sec

5. CONCLUSIONS

This paper introduces a software framework that allows quick flood simulation of multiple pipe failure events using 2D inundation models on modern desktop computer.

The framework is composed of multiple processes which: 1) reduce the computational cost of each event simulation by creating a minimal digital elevation model (DEM) for the sub-catchment of the pipe failure event; 2) execute a rapid 2D inundation model that uses cellular automata technique; 3) takes advantage of the large parallel computational power of the GPU of modern graphics cards.

The test case performed on a desktop machine shows that the framework can simulate around 500 pipe failure events on two citywide catchments in 88 minutes using a 2D model and high resolution terrains. Furthermore, the run times tests performed on four events show that the use of GPU computation to run the model produces run-times which are up to 9 times faster than multi-core CPU computation and that the use of reduces DEMs with expanding domain produces run-times which are up to 35 times faster than using full DEMs.

There are extra steps/features that can be added into the framework to further improve its performance and thus increasing the number of events that can be simulated in a short time. At the moment, the process that creates the reduced DEM for each event uses only minimally the information of the terrain around the pipe failure location. A future version could use the rolling ball algorithm to perform an initial terrain analysis for identifying all the possible ponds and flow paths of an

event as the base for creating the modelling domains of sub-catchments. This could further reduce the size of the DEM for each event with a slight increase of overhead. If multiple events are to be simulated at the same location, then the performance improvement obtained due to the use of the intial terrain analysis could vastly overtake the increase of overhead. By adding this improvement in the framework with executing the simulations in parallel with multiple graphics cards, could help the framework achieving the target of simulating thousand events per hour.

ACKNOWLEDGMENTS

The authors would like to acknowledge the funding provided by the U.K. Engineering and Physical Sciences Research Council, grant GR/J09796 (Cellular Automata Dual Drainage Simulation, CADDIES). The authors would like to thank the ICS consulting for supplying the pipe failure data and the Environment Agency for supplying the LiDAR datasets and the Ordnance Survey for the Mastermap dataset. Finally, the authors would like to acknowledge the support of NVIDIA Corporation with the donation of the Tesla K20c GPU used in this research and the support of Innovyze with the InfoWorks ICM 3.0 software license.

REFERENCES

[1] B. Ghimire, A. S. Chen, M. Guidolin, E. C. Keedwell, S. Djordjević, and D. A. Savić, 'Formulation of a fast 2D urban pluvial flood model using a cellular automata approach', *J. Hydroinformatics*, vol. 15, no. 3, p. 676, Jul. 2013.

[2] S. Wolfram, 'Cellular automata as models of complexity', *Nature*, vol. 311, pp. 419–424, Oct. 1984.

[3] M. Guidolin, A. Duncan, B. Ghimire, M. Gibson, E. Keedwell, A. S. Chen, S. Djordjevic, and D. Savic, 'CADDIES: A New Framework for Rapid Development of Parallel Cellular Automata Algorithms for Flood Simulation', presented at the 10th International Conference on Hydroinformatics (HIC 2012), Hamburg, Germany, 2012.

[4] L. Dagum and R. Menon, 'OpenMP: an industry standard API for shared-memory programming', *Comput. Sci. Eng. IEEE*, vol. 5, no. 1, pp. 46–55, 1998.

[5] A. Munshi and others, 'The OpenCL specification version 1.1', *Khronos OpenCL Work. Group*, 2011.

[6] O. Tange, 'GNU Parallel—the command-line power tool', *USENIX Mag.*, vol. 36, no. 1, pp. 42–47, 2011.

[7] Innovyze, 'InfoWorks ICM - Integrated Catchment Modeling', 2013. [Online]. Available: http://www.innovyze.com/products/infoworks_icm [Accessed: 02-Oct-2013].

[8] S. Néelz and G. (Garry) Pender, 'Benchmarking of 2D hydraulic modelling packages', Environment Agency, Horison House, Deanery Road, Bristol, BS1 9AH, Report, 2013.

ANALIZA PLAVLJENJA USLED PUCANJA VODOVODNIH CEVI U GRADOVIMA

M. GUIDOLIN, A. S. CHEN, E. C. KEEDWELL, Slobodan DJORDJEVIĆ, Dragan A. SAVIĆ
Centre for Water Systems, College of Engineering, Mathematics and Physical Sciences, University of Exeter, Harrison Building, North Park Road, Exeter, EX4 4QF, UK

Rezime

U ovom radu je prikazan postupak za brz proračun rizika od poplava u gradovima usled pucanja vodovodnih cevi. Korišćenjem digitalnog modela terena automatski se kreiraju ulazni podaci za vise različitih simulacija. Time je omogućeno efikasno i sveobuhvatno razmatranje čitavog niza potencijalnih mesta pucanja cevi, odnosno verovatnoće i posledica time izazvanog plavljenja u svim delovima grada. Vodne kompanije mogu da koriste ovaj postupak da identifikuju lokacije na kojima pucanje cevi moze da dovede do većih steta, čime se preciznije procenjuje rizik od ovog tipa poplava i omogućuje bolje planiranje upravljanja sistemima. Opisani postupak ilustrovan je sa nekoliko primera.

Ključne reči: Pucanje cevi, poplave u gradovima, ravansko modeliranje plavljenja, CADDIES, GPU

Redigovano 27.10.2017.